ZADATAK	STROJOPIS	DOM	SILUETA	COCI	STOGOVI	KAMIONI
input	standard input (*stdio*)					
output	standard output (*stdout*)					
time limit	1 second	0.8 seconds	0.2 seconds	1 second	1 second	3 seconds
memory limit	32 MB	32 MB	32 MB	32 MB	64 MB	64 MB
score	**50**	**80**	**100**	**120**	**140**	**160**
	total 650					

Proper typing is becoming an essential part of culture. If you are still not using **all ten fingers** for typing, you have to re-learn typing – then you will type faster and feel more comfortable and enjoyable.

There are a lot of web sites teaching proper typing. The following image depicts the basic principle; the keys needed to press with the same finger are of the same color. The yellow keys need to be pressed with the pinky, the blue ones with the ring finger, the green ones with the middle finger and the red ones with the index finger. Naturally, the left hand presses the left side of the keyboard (starting with keys 5, T, G, B to the left), the right hand presses the right side (starting with keys 6, Y, H, N to the right). Thumbs are responsible for space.

Please note: *the image depicts the US layout. For programming purposes, it is advised to switch to this layout because a lot of special characters, like* `[]`*, are easier to type. The US layout can be easily set on any operation system.*

Your task is to output how many times each finger, excluding thumbs, participated in typing the given string properly.

INPUT

The first and only line of input contains of a string consisting of at least one and at most fifty characters. The string **doesn't contain whitespaces** and consists **only of characters depicted on the image above**.

OUTPUT

The output must consist of eight lines, in each line one integer denoting the number of presses of each finger, excluding thumbs, observed from left to right.

SAMPLE TESTS

input	**input**	**input**
AON=BOO;	PRINT'NY'[NASLA]	VIDI,KO,JE,DOSA
output	**output**	**output**
1 0 0 1 1 0 3 2	2 1 0 2 4 1 1 5	1 1 3 1 1 6 2 0

In a retirement home, $\mathbf{N}$ senior citizens are watching television. The television programme consists of $\mathbf{M}$ channels denoted with numbers from 1 to $\mathbf{M}$. Each of the pensioners has their own favourite and hated TV channel.

If the television is currently displaying the hated channel of a certain pensioner, he will get up, slowly walk over to the TV set and change the channel to his favourite and get back to his comfortable chair. If there are multiple pensioners who hate the current channel, the youngest of them will get up (he's young, he doesn't mind) and the rest will remain seated.

Of course, after one change of channel, it is possible that another pensioner finds the new channel intolerable so he will change that channel. Given the fact that the pensioners are **stubborn**, this may continue indefinitely.

For the pensioners' favourite and hated channels and the initial channel on TV, determine the number of channel changes it takes for the pensioners to remain happy and seated.

INPUT

The first line of input contains three integers $\mathbf{N}$, $\mathbf{M}$ and $\mathbf{P}$ ($1 \leqslant \mathbf{N}, \mathbf{M} \leqslant 10^5$, $1 \leqslant \mathbf{P} \leqslant \mathbf{M}$), the number of pensioners, the number of TV channels and the initial channel displayed on TV.

Each of the following $\mathbf{N}$ lines contains two integers $\mathbf{a_i}$ and $\mathbf{b_i}$ ($1 \leqslant \mathbf{a_i}, \mathbf{b_i} \leqslant \mathbf{M}$, $\mathbf{a_i} \neq \mathbf{b_i}$), the favourite and hated channel of each pensioner.

The ordering of pensioners in the input is from the youngest to the oldest.

OUTPUT

The first and only line of output must contain the required number of channel changes. If the changes continue indefinitely, output -1.

SCORING

In test cases worth 50% of total points, it will hold $1 \leqslant \mathbf{N}, \mathbf{M} \leqslant 10^3$.

SAMPLE TESTS

input	**input**	**input**
3 4 2 1 2 2 3 3 2	3 3 1 1 2 2 3 3 1	4 5 2 1 3 2 3 3 2 5 1
output	**output**	**output**
1	-1	3

Clarification of the first example: Initially, channel 2 was on TV. This channel annoys the youngest and the oldest pensioner, so the youngest gets up enthusiastically and changes the channel so everyone can watch channel 1 together.

The main hero of this task, painter Vincent, spent a great deal of his youth travelling the world. Sights from numerous voyages have often been the inspiration for his, nowadays highly praised, works of art. On one occasion, Vincent found himself in a metropolis full of skyscrapers so he got down to work right away, intoxicated by the marvelous sight. For a number of reasons, incomprehensible to an average programmer, Vincent decided to paint only the **silhouettes** of the skyscrapers seen before him. Unfortunately, a week after he finished this masterpiece, the painting spontaneously caught fire.

In order to reconstruct the painting, Vincent sought help in all directions; architects provided him with the exact dimensions of the skyscrapers, physicists ignored air resistance, mathematicians mapped everything onto a plane and now it's your turn!

From your perspective, Vincent's skyscrapers are rectangles whose sides are parallel to coordinate axes and with one side that lies on the abscissa. Part of the abscissa on the image should be shown with the characters '*', the silhouettes of the skyscrapers with '#' and fill the rest of the image with '.'. The left edge of the image must begin with a skyscraper, whereas the right edge of the image must end with a skyscraper. Additionally, in order to verify the results the mathematicians got, output **the perimeter** of the given silhouette not calculating the sides that lie on the abscissa.

INPUT

The first line of input contains an integer $\mathbf{N}$ ($1 \leqslant \mathbf{N} \leqslant 10\,000$), the number of skyscrapers.

Each of the following $\mathbf{N}$ lines contains three integers $\mathbf{L_i}$, $\mathbf{R_i}$ and $\mathbf{H_i}$ ($1 \leqslant \mathbf{L_i}, \mathbf{R_i}, \mathbf{H_i} \leqslant 1,000$, $3 \leqslant \mathbf{R_i} - \mathbf{L_i} \leqslant 1,000$) that describe the position of the $\mathbf{i^{th}}$ skyscraper. That skyscraper, in a Cartesian coordinate system, is considered a rectangle with its lower left corner in ($\mathbf{L_i}$, 0) and upper right corner in ($\mathbf{R_i}$, $\mathbf{H_i}$).

OUTPUT

The first line of output must contain the perimeter of Vincent's silhouette.
The next $\mathbf{h}+1$ lines, where $\mathbf{h}+1$ is the height of the highest skyscraper, must contain Vincent's drawing as described in the task.

SCORING

In test cases worth 50% of total points, it will hold ($1 \leqslant \mathbf{N} \leqslant 100$) ($1 \leqslant \mathbf{L_i}, \mathbf{R_i}, \mathbf{H_i} \leqslant 100$).

If the programme outputs the correct perimeter, but the wrong image, it will get 40% of points for that test case.

If the programme outputs the wrong perimeter, but the correct image, it will get 60% of points for that test case.

If you can draw the image, but cannot calculate the perimeter, output a single integer in the first line and then output the image.

SAMPLE TESTS

input

```
3
1 5 4
7 11 3
9 13 5
```

output

```
28
........####
####....#..#
#..#..###..#
#..#..#....#
#..#..#....#
************
```

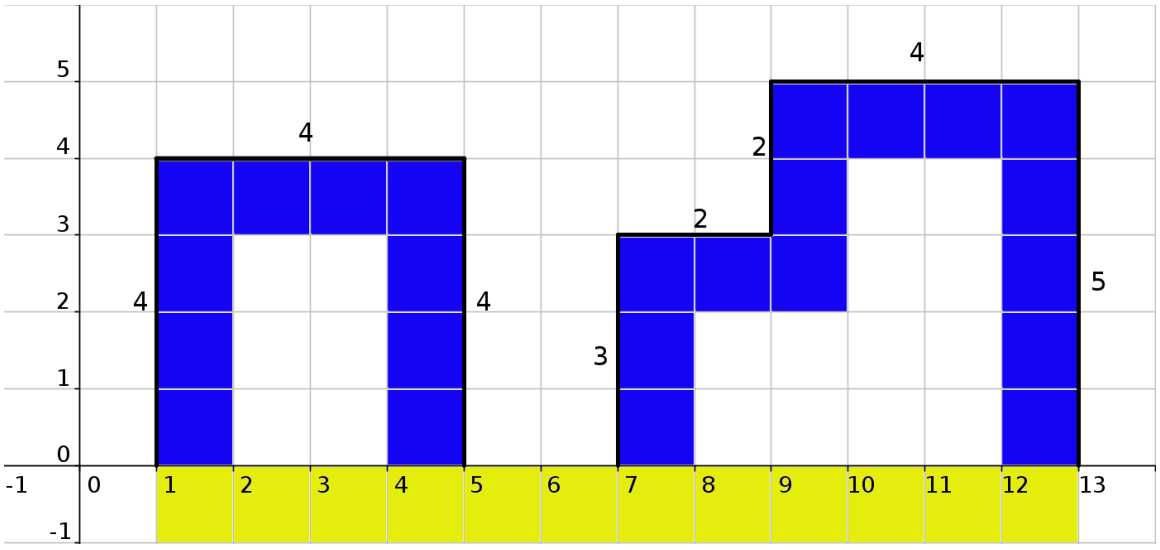
input

```
6
2 8 7
5 13 5
2 18 3
23 26 5
20 31 7
21 30 10
```

output

```
61
....................#########.
....................#.......#.
....................#.......#.
######.............##.......##
#....#.............#.........#
#....######........#.........#
#.........#.......#.........#
#.........######..#.........#
#...............#..#.........#
#...............#..#.........#
*****************************
```

Clarification of the first example: Blue color denotes the skyscrapers' silhouette (character '#'), whereas yellow is the part of abscissa located on Vincent's painting (character '*').



The 3rd round of COCI is already here! In order to bet on predict the scores, we have assumed the following:

- If contestant A scored strictly more points than contestant B in **each** of the first two rounds, then in the third round A will score **at least an equal amount** of points as B.

Of course, in each round (including this one, the 3rd one) it is possible to score from 0 to 650 points. On the **total ranking list**, contestants are sorted descending according to the **sum of points from all three rounds**. The contestants with an equal sum share the same place and the next contestant gets the realistic following place. For example, contestants with sums equal to 1000, 1000, 900, 900 and 800 points win places 1., 1., 3., 3. and 5., respectively.

For each of the **N** contestants, we know the number of points scored in the first and second round. Given the aforementioned assumption, determine the highest and lowest place each contestant can get on **the total ranking list** after three rounds of COCI.

INPUT

The first line of input contains an integer **N** ($1 \leqslant \mathbf{N} \leqslant 500\,000$), the number of contestants.

Each of the following **N** lines contains two integers from the interval $[0, 650]$: the number of points each contestant won in the first and second round.

OUTPUT

For each contestant, in the order given in the input, output two integers per line: the required highest and lowest place they can get on the total ranking list.

SAMPLE TESTS

input

```
5
250 180
250 132
220 123
132 194
220 105
```

output

```
1 3
1 3
3 5
1 5
3 5
```

input

```
10
650 550
550 554
560 512
610 460
610 456
650 392
580 436
650 366
520 456
490 456
```

output

```
1 4
1 8
2 8
2 7
2 9
1 10
4 10
1 10
5 10
5 10
```

Mirko is playing with stacks. In the beginning of the game, he has an empty stack denoted with number 0. In the $\mathbf{i}^{\mathbf{th}}$ step of the game he will choose an existing stack denoted with $\mathbf{v}$, copy it and do one of the following actions:

a. place number $\mathbf{i}$ on top of the new stack

b. remove the number from the top of the new stack

c. choose another stack denoted with $\mathbf{w}$ and count how many different numbers exist that are in the new stack and in the stack denoted with $\mathbf{w}$

The newly created stack is denoted with $\mathbf{i}$.

Mirko doesn't like to work with stacks so he wants you to write a programme that will do it for him. For each operation of type b output the number removed from stack and for each operation of type c count the required numbers and output how many of them there are.

INPUT

The first line of input contains the integer $\mathbf{N}$ ($1 \leqslant \mathbf{N} \leqslant 300\,000$), the number of steps in Mirko's game.

The steps of the game are chronologically denoted with the first $\mathbf{N}$ integers.

The $\mathbf{i}^{\mathbf{th}}$ of the following $\mathbf{N}$ lines contains the description of the $\mathbf{i}^{\mathbf{th}}$ step of the game in one of the following three forms:

- "a v" for operation of type a.
- "b v" for operation of type b.
- "c v w" for operation of type c.

The first character in the line denotes the type of operation and the following one or two denote the accompanying stack labels that will always be integers from the interval $[0, \mathbf{i} - 1]$.

For each operation of type b, the stack we're removing the element from will not be empty.

OUTPUT

For each operation type b or c output the required number, each in their own line, in the order the operations were given in the input.

SAMPLE TESTS

input

```
5
a 0
a 1
b 2
c 2 3
b 4
```

output

```
2
1
2
```

input

```
11
a 0
a 1
a 2
a 3
a 2
c 4 5
a 5
a 6
c 8 7
b 8
b 8
```

output

```
2
2
8
8
```

Clarification of the first example: In the beginning, we have the stack $S_0 = \{\}$. In the first step, we copy S_0 and place number 1 on top, so $S_1 = \{1\}$. In the second step, we copy S_1 and place 2 on top of it, $S_2 = \{1, 2\}$. In the third step we copy S_2 and remove number 2 from it, $S_3 = \{1\}$. In the fourth step we copy S_2 and denote the copy with S_4, then count the numbers appearing in the newly created stack S_4 and stack S_3, the only such number is number 1 so the solution is 1. In the fifth step we copy S_4 and remove number 2 from it, $S_5 = \{1\}$.

We are observing the movement of **N** trucks on the road. The road can be represented as a number line. The points on the line denoting cities are integers. The cities are denoted with the number of the corresponding point on the line.

All trucks are moving with **the same speed** and **no single truck is standing still at any given moment**. Each truck takes 1 minute to pass the distance between two adjacent cities.

You are given the **route** of each truck. All of the trucks start their route **at the same initial moment**.

The route is given as an array of **k** cities: $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_k}$. The truck starts from city $\mathbf{A_1}$ and drives to city $\mathbf{A_2}$, then turns and drives to city $\mathbf{A_3}$ and so on. Given the fact that the truck turns, it will hold:

$$\mathbf{A_1} < \mathbf{A_2} > \mathbf{A_3} < \mathbf{A_4} > ... \text{ or } \mathbf{A_1} > \mathbf{A_2} < \mathbf{A_3} > \mathbf{A_4} < ...$$

The time it takes for the truck to turn is negligible.

One possible route is 2, 5, 1, 7. The truck is at city number 2 initially, 3 minutes after departure it arrives to city number 5. It turns and continues towards the city number 1 in which it arrives 7 minutes after departure. It turns again and drives towards city number 7 in which it arrives at moment 13.

After the truck completes his route, aliens appear and take it away in their space rocket.

For some pairs of trucks, we want to know the number of times they encountered each other on the road. In other words, how many times they appeared to be on the same position (the position they encountered doesn't need to be an integer; i.e. they could've encountered at position 2.5).

Write a programme that will, for a given number of trucks **N** and their routes and for **M** pairs of trucks, determine the number of encounters for each pair.

Please note: each pair of trucks we want to know the number of encounters for, it will hold:

- they won't be at the same place in the moment when one of them (or both) are being taken away by aliens
- they won't be at the same place in the initial moment or in the moment when one of them (or both) are turning

The upper statement **won't hold** for all pairs of trucks, but **only the pairs** we want to know the number of encounters for.

INPUT

The first line of input contains the integers **N** and **M** ($1 \leqslant \mathbf{N} \leqslant 10^5$, $1 \leqslant \mathbf{M} \leqslant 10^5$), the number of trucks and the number of pairs of trucks we want to know the number of encounters for.

The $\mathbf{i^{th}}$ of the following **N** lines contains the description of the route of the $\mathbf{i^{th}}$ truck.

The first integer in the line, $\mathbf{K_i}$ ($2 \leqslant \mathbf{K_i} \leqslant 3 \cdot 10^5$) represents the number of cities on the truck's route. After it $\mathbf{K_i}$ integers follow, $\mathbf{A_j}$ ($1 \leqslant \mathbf{A_j} \leqslant 10^9$), the ordinal numbers of the cities on the truck's route given in the order which the truck visits them.

The sum of routes of all the trucks won't exceed $3 \cdot 10^5$.

Each of the following **M** lines contains two integers ($\mathbf{a_i}, \mathbf{b_i}$), the ordinal numbers of the trucks we want to know the number of encounters for.

OUTPUT

Output **M** lines. The $\mathbf{i^{th}}$ line must contain the number of encounters of the $\mathbf{i^{th}}$ pair of trucks from the input.

SCORING

In the test cases worth 50% of total points, it will hold $\mathbf{N} \leqslant 10^2$, $\mathbf{K_i} \leqslant 10^3$, $\mathbf{M} \leqslant 10^3$.

SAMPLE TESTS

input

```
3 3
3 1 3 1
2 2 1
3 3 1 3
1 2
2 3
3 1
```

output

```
1
0
2
```

input

```
2 1
4 1 6 3 6
7 3 4 2 6 5 6 1
1 2
```

output

```
3
```

input

```
3 4
3 1 4 2
4 3 4 2 4
3 4 1 3
1 2
2 3
3 1
1 3
```

output

```
2
1
2
2
```